A STORAGE SYSTEM AND METHOD FOR PRESTAGING DATA IN A CACHE FOR IMPROVED PERFORMANCE

Technical Field

[001] This invention relates to computer storage systems, and more particularly to a storage system and method for prestaging data in a cache memory based on relative changes in data access frequency and the effectiveness of previously prestaged data.

Background of the Invention

[002] Storage systems are one type of auxiliary computing storage devices where each system includes a large number of disk drive units. Large enterprise-level storage systems must have relatively high performance characteristics to meet the performance of high-performing application servers, file servers and database systems supported by the storage systems. In these storage systems, reading data from and writing data to the disk drive units are fairly time-consuming because of the lengthy mechanical operations within the disk drives. Examples of these operations include the arm movement in a disk drive or the rotational delay of the disk associated with getting a read/write head into a reading position or a writing position. To provide fast access to frequently accessed data, cache memories are typically used in the storage systems to temporarily hold this data. Since the read latency for data from a cache memory is less than that for a disk drive unit, the presence of the cache memory significantly improves the overall throughput of a storage system.

[003] To further reduce the read latency, storage systems also use prestage operations to retrieve data from a disk drive into a cache before the data is retrieved by the next host I/O request. This can be done by the host issuing a prestage command, such as the extent channel command used in mainframe programs to indicate that a sequential access will take place, to the storage system

to move data to cache. Alternatively, the storage system anticipates the next host I/O request and retrieves the data without any special hint or command from host application.

[004] Since a cache memory has a much higher cost per byte than the data storage, its size is significantly smaller than the total storage. Resource management of the cache is typically done through a Least-Recently-Used (LRU) algorithm. The time duration since the last use of the data in the cache is an indication of its frequency of use. Data stored in the cache memory is aged from the point of time of its last use. Due to the limited capacity of the cache, data is continuously removed from the cache's address space as it becomes the least recently used data. While infrequently accessed data periodically enters the cache, it will tend to age and fall out of cache under the Least-Recently-Used algorithm.

[005] Prior art prestage algorithms assume that data that is sequentially accessed is likely to be accessed in close temporal order or vice versa. In current storage systems, prestage algorithms use metadata to identify a perfect sequentially access I/O pattern to signal to the systems that a prestage operation is necessary. As an example, assume that LBAx, LBAx+y, LBAx+2y, LBAx+3y and so on are contiguous chunks of logical block addresses (LBAs) on a disk drive unit. There are different ways how the host can access the data, which affect the prestage algorithms as a result.

[006] If the order of accesses is LBAx, LBAx+y, LBAx+2y and LBAx+3y, the storage system will prestage LBAx+2y and LBAx+3y only after LBAx, LBAx+y and more perfectly sequential data are accessed one following the other.

[007] If the order of accesses is LBAx, LBAx+2y, LBAx+y and LBAx+3y, the storage system will fail to prestage LBAx+3y after LBAx is accessed.

[008] If the order of accesses is LBAx, LBAx+2y, LBAx+4y, LBA+6y and so on, the storage system will also fail to issue any prestage in the region at all.

[009] The size of the metadata associated with perfectly sequential access data is limited to the size of the memory in the storage systems. This configuration

limits the overall accuracy and comprehensiveness of the recorded metadata. Since the gap between the size of the metadata memory in the storage system and its total storage capacity is growing rapidly, the recorded metadata does not accurately represent the I/O behavior.

[010] United Stated patent 6,44,697 describes a method for prestaging data into a cache to prepare for data transfer operations. The method determines addressable locations of the data to be cached and generates a data structure capable of indicating contiguous and non-contiguous addressable locations in the storage system. A prestage command then causes the data at the addressable locations in the data structure to be prestaged into the cache. This prestage method does not take into consideration relative changes in the data access frequency and relative improvements in previous prestage operations.

[011] United States patent 6,260,115 describes a method for prestaging data in a storage system by detecting a sequential access pattern and then prestaging a number of data tracks ahead of the current request based on the available storage. Data accesses are maintained in a list in the most-recently-used order from which sequential access patterns are detected. A key disadvantage of this method is that its benefits are realized only if there is a perfect sequentiality in the I/O stream.

[012] Therefore, there remains a need for a storage system and method for efficiently prestaging data without the drawbacks of the prior art methods described above.

Summary of the Invention

[013] It is an object of the present invention to provide a storage system capable of prestaging data in a cache memory based on relative changes in the frequency of access to data in a region and in the effectiveness of prior prestaging operations.

[014] It is another object of the invention to provide a method for prestaging data in a cache memory based on relative changes in the frequency of access to

data in a region and in the effectiveness of prior prestaging operations.

[015] It is yet another object of the invention to provide a storage system and method for prestaging data in which relative changes in the frequency of data access are determined based on access statistics of the storage regions. Access statistics preferably include data location, I/O size and access frequency.

[016] It is still another object of the invention to provide a storage system and method for prestaging data in which relative changes in the effectiveness of previous prestage operations are determined based on information concerning the prestaged data and host requests for the prestaged data.

[017] To achieve these and other objects, the invention provides a method for prestaging data in a storage system having a cache that includes: (a) determining a relative change in the frequency of data access for a storage region in the system; (b) determining a relative change in the effectiveness of previous prestage operations; and (c) determining whether to prestage data in the cache and the amount of data to prestage based on the determined relative change in access frequency, the determined relative change in the effectiveness and the size of last I/O access. The determination of a relative change in the data access frequency preferably includes the steps of: (a) maintaining statistics on data access to the region such as data location, I/O size and access frequency; and (b) comparing the statistics of recent I/O requests to the maintained statistics to determine relative changes in the access frequency for the region.

[018] The determination of a relative change in the effectiveness of previous prestage operations preferably includes: (a) recording the number of previous prestaging operations of data for the region; (b) recording the number of I/O requests for data that has been prestaged for the region and is present in the cache; and (c) determining the relative change in the effectiveness by dividing the number of host I/O requests for previously prestaged data present in the region during a time period by the number of previous prestage operations for the region during the same period. If the relative change in the frequency of data access and

the relative change in the effectiveness are both increasing, then data for the region is prestaged aggressively, i.e., a relatively large amount of data is prestaged. If either the relative change in the frequency of data access or the relative change in the effectiveness is increasing and the other measure is decreasing, then data for the region is prestaged only moderately. If both of these measures are decreasing, then data should be prestaged very minimally since there is little benefit in doing so.

[019] Additional objects and advantages of the present invention will be set forth in the description which follows, and in part will be obvious from the description and the accompanying drawing, or may be learned from the practice of this invention.

Brief Description of the Drawing

[020] Figure 1 is a block diagram illustrating a storage system environment in which data might be prestaged in accordance with the invention.

[021] Figure 2 is a block diagram illustrating different levels of the storage system in which the prestaging operations might be implemented in accordance with the invention.

[022] Figure 3 is a flow chart showing a prior art method for prestaging data in a storage system.

[023] Figure 4 is a flow chart showing a high level process for prestaging data in accordance with the invention.

[024] Figure 5 is a flow chart showing a preferred embodiment of the process for prestaging data in accordance with the invention.

[025] Figure 6 is a flow chart showing a preferred embodiment of the process for including information on the effectiveness of previous prestaging and relative changes in data access frequency to determined future data prestaging operations.

[026] Figure 7 is a chart showing a preferred policy for determining the amount of data for prestaging depending on a region's heat and the effectiveness of prior data prestaging.

Description of the Preferred Embodiments

[027] The invention will be described primarily as a method for prestaging data in a storage system based on relative changes in data access frequency and the effectiveness of prior prestaging operations. However, persons skilled in the art will recognize that an apparatus, such as a data processing system, including a CPU, memory, I/O, program storage, a connecting bus, and other appropriate components, could be programmed or otherwise designed to facilitate the practice of the method of the invention. Such a system would include appropriate program means for executing the operations of the invention.

[028] Also, an article of manufacture, such as a pre-recorded disk or other similar computer program product, for use with a data processing system, could include a storage medium and program means recorded thereon for directing the data processing system to facilitate the practice of the method of the invention. Such apparatus and articles of manufacture also fall within the spirit and scope of the invention.

[029] The following key terms are now defined to facilitate the description of the invention that follows.

[030] Track - A logical and contiguous collection of storage units.

[031] Region - A collection of multiple tracks. Prestaging decisions are made in each individual region. A region within the physical volume can be fixed or variable sized.

[032] Heat - The "heat" is a measurement of frequency of I/O access from host. A hot region indicates that a lot of I/O accesses were recently made regarding the region. Similarly, a cold region indicates that there hasn't been much

I/O access in the region recently.

[033] Effectiveness - The "effectiveness" is a measurement of the benefits resulting from past prestaging operations. It can be viewed as the return of the investment made in prestaging the data so far. The return on the prestaging invention is good if the prestaged data in the cache is requested by the host. This results in a lower latency of the host I/O operation. On the other hand, if the prestaged data is demoted from the cache without having a host read hit, then it has wasted system resources, such as CPU power, memory space and I/O bandwidth, in both staging the data and in keeping the data resident in cache. The benefits of prior prestaging decisions can be evaluated by determining whether the prestaged data has been accessed by the host or not.

[034] Thermal Sensor - A component to gauge heat of a region.

[035] Prestage Effectiveness Meter - A component to gauge the effectiveness of previous prestaging of data from a region.

[036] Prestage Range - A range of LBAs used for the next prestage operation within a region.

[037] Figure 1 is a block diagram representing a high-level view of a storage system in which data might be prestaged in accordance with the invention. A host computer system 10 is typically connected to a storage controller 12 through a network 11. As an example, the storage controller 12 may be a SAN Volume Controller manufactured by IBM Corporation and the network 11 may be a Fiber Channel storage area network. The storage controller 12 is connected to a storage disk 14 through a network 13. The storage disk 14 may be a disk array FAST-T600 offered by IBM Corporation and the network 13 may be another storage area network. The host computer 10 accommodates many software components, including an application program 15 which sends input and output operations to the storage controller 12. The application program 15 must provide the storage controller 12 with details of a data request such as the kind of operation involved, the storage volume that the operation is for, the logical block address of the first

block of the data, and the size of the data to be processed. An operation might be a read of data from the disk 14 (an output operation) or a write of data to the disk 14 (an input operation). The storage controller 12 receives the input or output operations and processes them accordingly. This processing may or may not involve a staging operation or a destaging operation on the storage disk 14. On receipt of responses from the disk 14, the storage controller 12 returns the completion status of the operation as well as any applicable data to the application 15. The host computer 10 and the storage controller 12 may communicate with each other using a network protocol that is suitable for the network 11, e.g., the Fiber Channel protocol. The storage controller 12 and the disk 14 may communicate with each other using a network protocol applicable to the network 13 such as the Fiber Channel protocol. Data from the storage disk 14 might be prestaged in a high-speed cache memory typically implemented using volatile memory. The cache may reside anywhere in the input/output path between the application 15 on the host computer 10 and the disk 14.

[038] Figure 2 illustrates an expanded storage environment in which there are two host computers 20 that are connected to two storage controllers 22 through a network 21. The storage controllers 22 access data stored on a disk array 24 through a storage network 23. As shown in Figure 2, one or more cache memories 26 might be implemented in any of the host computers 20, the storage controllers 22 or the data array 24. Since the storage controllers 20 have knowledge of the data requests issued by an application program on the host systems 20, they may prestage data from the disk array 24 in anticipation of subsequent requests for data before such requests actually arrive, regardless of where a cache 26 resides in the I/O path.

[039] I/O retrieval time must be shortened to bridge the gap between processor performance and storage performance. When an application on a host system 21 requests data from storage, a storage controller 22 in the I/O path attempts to find the data in the cache 26. If this lookup fails, the storage controller 22 stages the data from the disk array 24 into the cache 26. This is a time

consuming operation and thus, data cache misses are costly to the application. Since the measure of cache effectiveness is its miss ratio, and its complement - the hit ratio, cache performance can be improved by prestaging data before the data is requested by the host application. Such a prestaging takes advantage of the general spatial locality in data streams (i.e., perfect patterns) and moves data into the cache to benefit future I/O accesses. Perfect sequential access patterns are I/O access sequences that match the logical order of the data blocks in the system. Imperfect patterns are I/O sequences that are random within the region. The invention detects both perfect and imperfect sequential access patterns that can benefit from such prestaging.

[040] Figure 3 is a flow chart illustrating a typical prior art method for prestaging data in a storage system. At step 30, the storage controller accepts a data read request from a host application. At step 31, if the size of the requested data is larger than a track, the host request is logically divided into more than one track-sized requests and each of these requests is processed separately. At step 32, for each track, the storage controller checks to see if data is present in the cache. If data is present in the cache (a cache hit), then the storage controller retrieves the requested data from the cache at step 34. At step 35, the cache returns the retrieved track data to the host computer while maintaining the order with respect to the other tracks of the same read operation. If data is not in the cache (a cache miss), then the storage controller submits the request to the lower layers in the storage subsystem to obtain the data from the disk and stages this data in the cache, at step 33. Data is similarly returned to the host application in order at step 35. The host request operation is completed at step 36.

[041] Figure 4 is a flowchart depicting a high-level process for prestaging data in accordance with the invention. At step 40, the storage controller accepts a data request from the host. The data request is divided into logical track sizes at step 41. At step 42, the storage system checks whether the requested data for each track size (e.g., 32 Kbytes) is present in the cache. If data is already in the cache, the storage controller fetches the data from the cache at step 43.

Otherwise, it retrieves data from the disk at step 44. The retrieved data is next returned to the host at step 45. At step 46, the storage controller determines whether to prestage data from the end of the last request onward into the cache. If data is not going to be prestaged, then the host request processing ends at step 47. Otherwise, the storage controller determines the amount of data to be prestaged at step 48. This determination is based on the region's heat, the effectiveness from prior prestaging activities and the I/O size. The storage controller then initiates the prestage operation at step 49. The prestage operation is treated as a host request for data where the amount of data to be prestaged is first divided into logical track sizes at step 41.

[042] Figure 5 is a flowchart showing the details of a preferred process for data prestaging in accordance with the invention. At step 50, a host request for data is received by the storage controller. At step 51, the request is divided into multiple logical track sizes. If the request is a data read operation, then a Cache Statistics Counter is incremented at step 52. The Cache Statistics Counter indicates the number of track read I/O operations. At step 53, the storage controller determines the logical region that the requested tracks belong to based on the volume ID and the logical address of the first block of the requested data. If the request is a read operation, then a Region Statistics Counter is incremented at step 54. This counter indicates the number of track-size reads in the logical region. At step 55, for each track, the storage controller determines whether the track data is present in the cache. If data is present in the cache (i.e., a cache hit), then the controller determines, at step 56, whether this data was in the cache due to a previous prestage operation. If data has previously been prestaged, then a Region Prestage Hit Counter is incremented at step 57. The Region Prestage Hit Counter indicates the number of data tracks that have been found in the cache due to previous prestage operations. The requested data is then retrieved from the cache at step 58. If the track data is not present in the cache, then the controller staged this data in the cache at step 59. The retrieved data is returned to the host system in the order of the track sizes at step 60.

[043] At step 61, the storage controller determines whether to prestage in the cache data from the end of the last request onward. This determination is based on previous data accesses (i.e., heat) and the effectiveness of prior prestage operations. The heat of a logical region indicates the frequency of i/O accesses in the region by the host. If data is not to be prestaged at this point, then the processing of the host request ends at step 63. Otherwise, the storage controller determines the amount of data to be prestaged at step 62. This determination is based on the heat of the region, how effective the prestaging has been, and the size of the I/O request. Further details on the determination of the amount of data to be prestaged are described below in reference to Figure 7. The storage controller next increments a Region Prestage Start Counter at step 64. This counter indicates the number of prestage operations that have been initiated for the current logical region. The controller then initiates the prestage operation at step 65. The prestage operation is treated as a host request for data where the amount of data to be prestaged is first divided into logical track sizes at step 51.

[044] The invention uses information on the region's heat and the success of prior prestage operations to effectively prestage in the cache for anticipated future I/O requests. Figure 6 is a block diagram representing the processing of this information to generate future prestaging operations. In the preferred embodiment of the invention, there is conceptually a Thermal Sensor 66 for receiving information about the numbers of host data requests for different regions in the storage system. Based on this information, the Thermal Sensor 66 calculates the relative heat of a particular region, which is referred to as Δ heat /Δ time. In the preferred embodiment of the invention, the Thermal Sensor 66 calculates the region's relative heat based on the number of read operations for the region and the number of read operations for all regions in the system within a time period. The time period can be calculated in a number of ways: it can be a fixed time period when the heat of the particular region is sampled, it can be the period between n consecutive host I/O operations into this region, or it can be based on a predefined event.

[045] For example, let Tn represent the n-th time slice. At time T1, the number of read operations into the region, i.e., the value of the Region Statistics Counter, is x. The total number of read operations into all the regions, i.e., the value of the Cache Statistics Counter, is y. Therefore, Δ heat /Δ time = (x- 0)/ (y- 0) = x/y. At time T2, the number of read operations into the region is x'. The total number of read operations into all regions is y'. Therefore, Δ heat / Δ time = (x'-x) / (y'-y). A large value of Δ heat / Δ time suggests that an I/O stream is active on that region. By the principles of locality, a "hot" region would be a good candidate for a prestage operation.

[046] Figure 6 also shows a Prestage Effectiveness Meter 67 which receives information on the number of useful prestage operations and the total number of prestage operations. The Prestage Effectiveness Meter 67 calculates the effectiveness of previous prestaging of data which is referred to in the present description as Δ effectiveness /Δ time. The meter 67 uses the number of beneficial prestages (i.e., those that gets host read hits) versus the total number of prestages in the region within the time period, to measure the benefit of the previous prestaging. The first number is the value of the Region Prestage Hit Counter while the second number is the value of the Region Prestage Start Region. The time period can be calculated in a number of ways: it can be a fixed time period when the heat of regions is sampled, it can be the period between n consecutive host I/O operations into the region, or it can be based on a predefined event.

[047] For example, let Tn represent the n-th time slice. At time T1, let the number of prestaged tracks in the region that got host read hits be x. The total number of prestage operations in that region is y. Therefore, Δ effectiveness /Δ time = (x- 0)/ (y- 0) = x/y. At time T2, let the number of prestaged tracks in the region that got host read hits be x'. The total number of prestage operations in that region is y'. Therefore, Δ effectiveness / Δ time = (x'-x) / (y'-y). A large value of Δ effectiveness / Δ time indicates that the previous prestaging of data was effective and suggests that the system should continue to prestage data for this region. A low value indicates that prior prestage decisions have been ineffective and a

corrective action must be taken regarding future prestaging operations. A prestage operation is effective when the first host request after the prestage operation accesses the prestaged data. This operation causes an increase in the prestage effectiveness. However, further host accesses to the prestaged data before it has been demoted from the cache do not increase regional effectiveness as the prior host access would have cached the data if it had not been a cache hit.

[048] The information generated by the Thermal Sensor 66 and Prestage Effectiveness Meter 67, i.e., Δ heat /Δ time and Δ effectiveness /Δ time, is sent to an I/O Filter 68 to determine whether data should be prestaged for this region and if so the size of the I/O request to be prestaged. The policy for determining the future prestaging is preferably based on static values of the heat and effectiveness described above. The policy might compare the current heat and effectiveness with a previous heat gradient and effectiveness gradient to decide whether the region is "hotter" than before. The I/O filter 66 can then use this information to determine whether prestaging will be beneficial or not. Depending on the host I/O size 69 and the outputs of the Thermal Sensor 66 and the Prestage Effectiveness Meter 67, the I/O Filter 68 decides whether to generate a prestage request and the size of the I/O request.

[049] Figure 7 is a chart illustrating the relationship between a region's heat and the effectiveness of prestaging. Ht represents a static heat threshold. Et1 is a static lower effectiveness threshold. Et2 is a static higher effectiveness threshold. The values of Ht, Et1 and Et2 can be tuned for any give system. The decision whether to prestage data or not, given the values of heat and effectiveness, can be based on the kind of workloads currently handled by the system. For example, deciding not to prestage on heat values below the heat threshold except in instances when the prestage effectiveness is relatively high, prevents prestage operations for workloads that do not exhibit sufficient spatial locality.

[050] Data prestaging consumes system resources, such as memory and computing time, and can even compete for these resources with host I/O operations. It is important that all resources consumed by prestage operations

eventually have a overall positive impact on system behavior and also that the downside for any kind of workload or access pattern must be minimized. Preferably, the system resources are partitioned such that prestage operations would have accessed to only a percentage of the total available resources. This establishes a bound the total overall investment of system resource for the purpose of prestaging. Efficient and effective resource management also requires that any resource used by bad prestage investment, should be reclaimed by the system as soon as the information of bad decision is known. Memory resources used for maintaining statistics per region, can grow very big depending on the total storage size. Resource management of the statistics table can follow a simple replacement algorithm of statistics entries. The replacement algorithm can store some statistic table entries on storage based on most recent access and/or frequency of use so that at any given time, only a limited amount of memory is actually paged in to keep statistics.

[051] While the present invention has been particularly shown and described with reference to the preferred embodiment, it will be understood by those skilled in the art that various changes in form and detail may be made without departing from the spirit and scope of the invention. Accordingly, the disclosed invention is to be considered merely as illustrative and limited in scope only as specified in the appended claims.